
AutoLayout

강사 주영민

Storyboard 사용하기

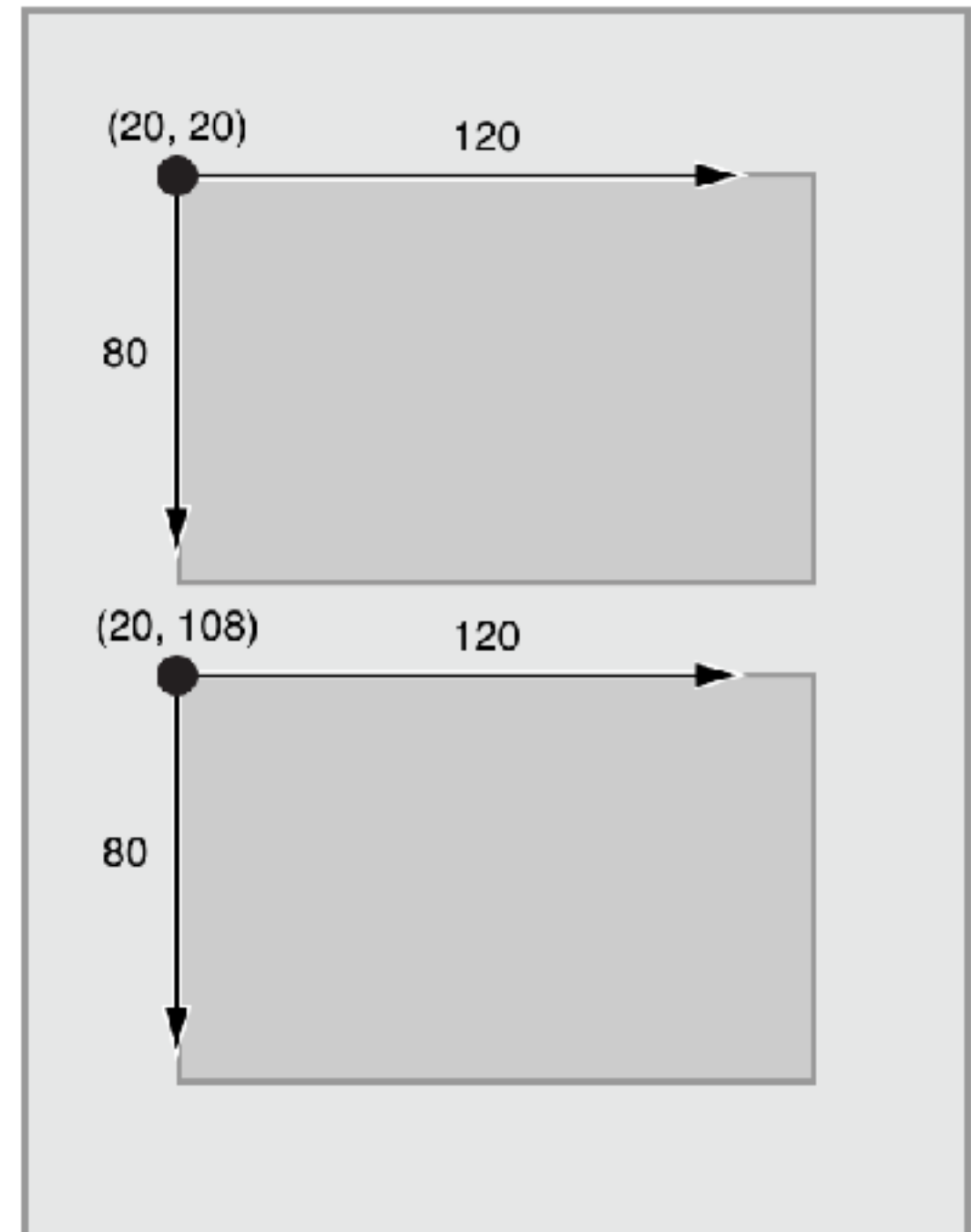
- IBOutlet : UI 아울렛 연결
- IBAction : UI Action 추가

AutoLayout

- Auto Layout dynamically calculates the size and position of all the views in your view hierarchy, based on constraints placed on those views

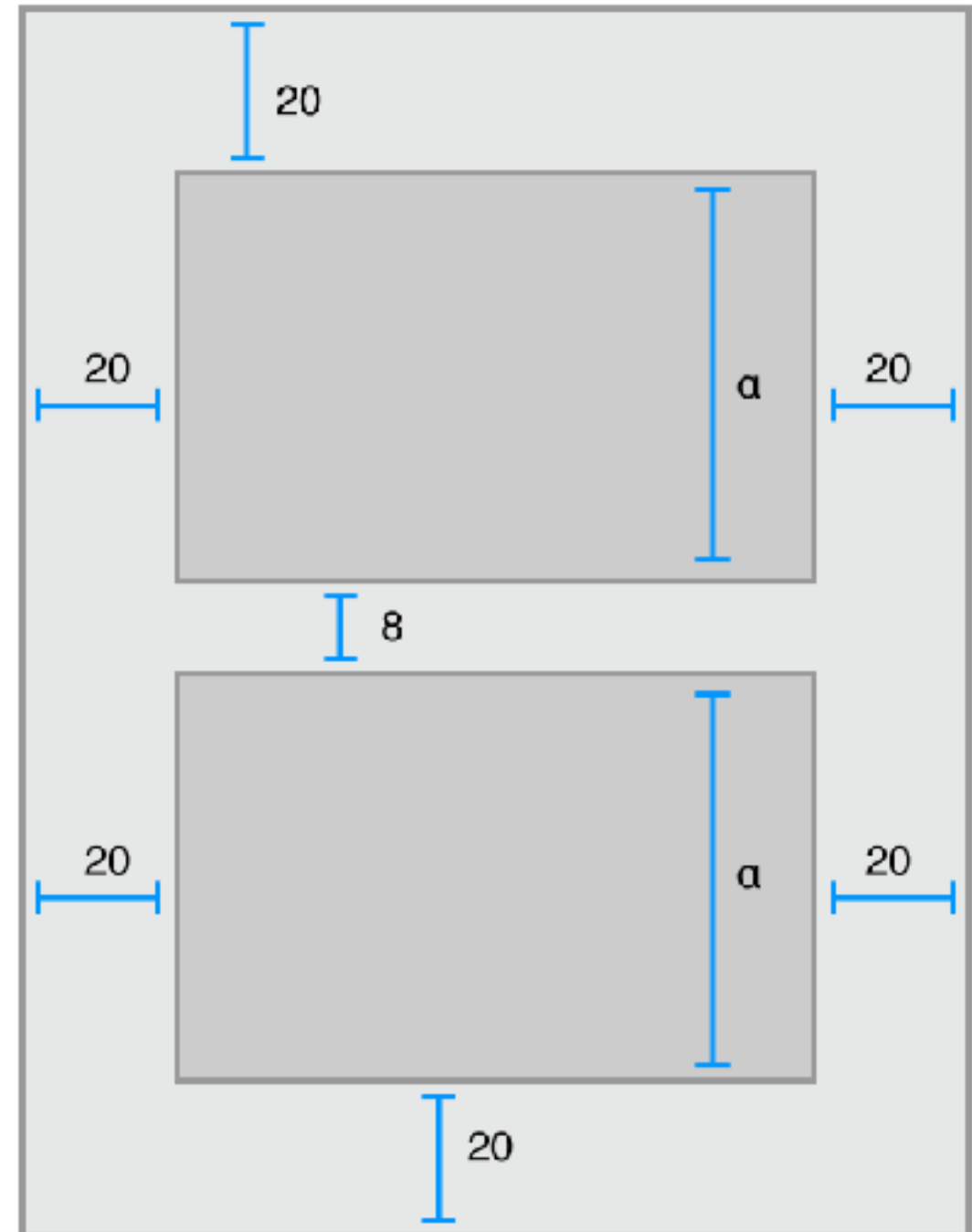
Auto Layout VS Frame-Based Layout

- Frame-Based Layout



Auto Layout VS Frame-Based Layout

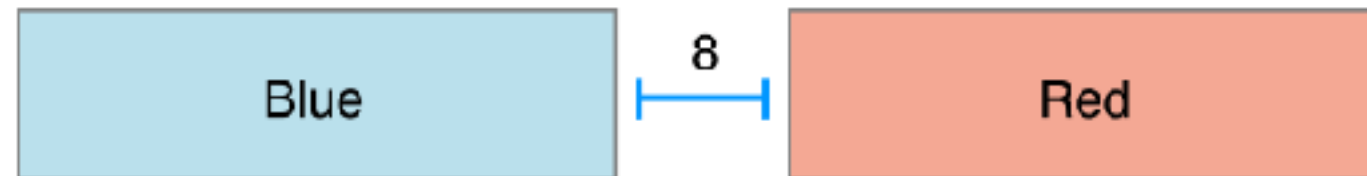
- Auto Layout



Constraint

- 각 뷰의 거리, 길이, 위치 등을 표현하기 위한 제약

Constraint 해부



$$\underbrace{\text{RedView.Leading}}_{\text{Item 1}} = \underbrace{1.0}_{\text{Multiplier}} \times \underbrace{\text{BlueView.trailing}}_{\text{Item 2}} + \underbrace{8.0}_{\text{Constant}}$$

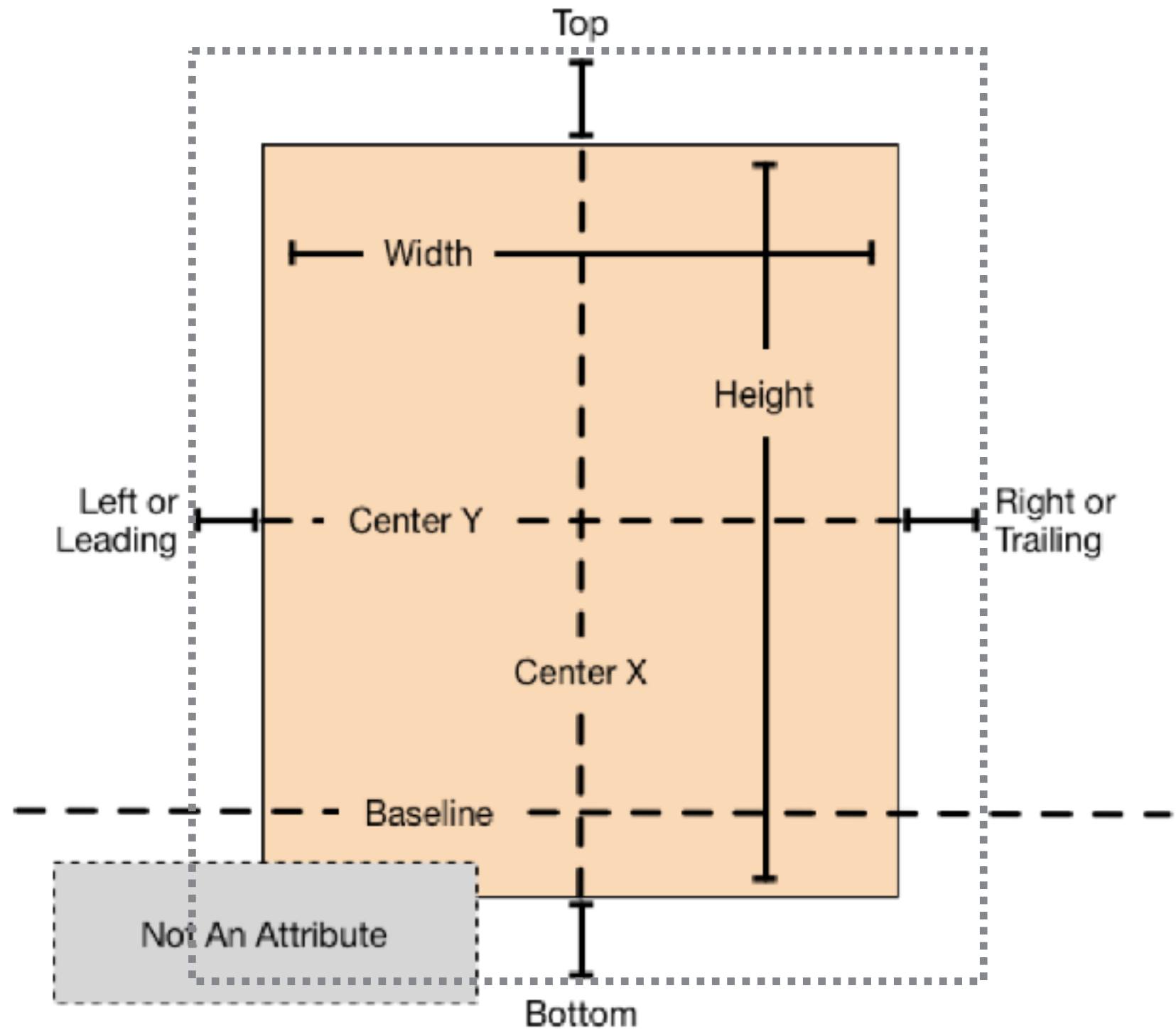
Relationship

Attribute 2

Attribute 1

Attribute

- Size attributes
 - ✓ width
 - ✓ height
- Location attributes
 - ✓ Leading
 - ✓ Trailing
 - ✓ Top
 - ✓ Bottom
 - ✓ Vertical
 - ✓ Horizontal



Multiplier

- 비율을 통한 레이아웃 설정을 위한 속성

Constant

- 일정한 간격을 유지하기 위한 속성

Constraint 공식

대상 View의 Attribute는 기준View의 Attribute X 비율 +간격이다.

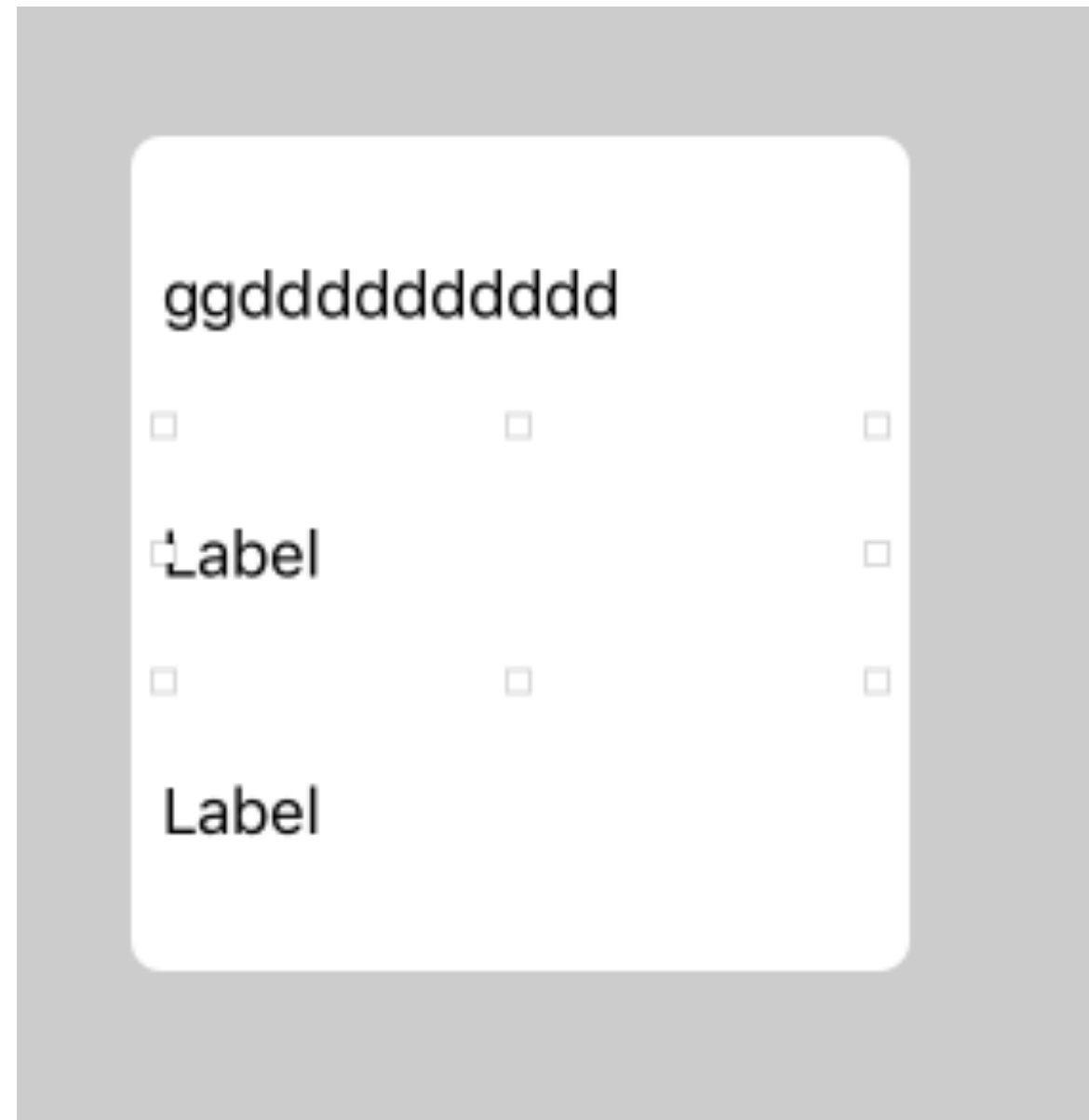
$$\text{Item1.Attribute} = \text{비율} \times \text{Item2.Attribute} + \text{간격}$$

제한 사항

- 우리는 AutoLayout을 적용하기 위해, 두 View의 거리, 정렬, 두 뷰간의 상대적 크기, 또는 비율등의 제약을 설정할 것이다. 하지만 이러한 설정들이 모두 호환가능하진 않다.
 - ✓ You cannot constrain a size attribute to a location attribute.
 - ✓ You cannot assign constant values to location attributes.
 - ✓ You cannot use a nonidentity multiplier (a value other than 1.0) with location attributes.
 - ✓ For location attributes, you cannot constrain vertical attributes to horizontal attributes.
 - ✓ For location attributes, you cannot constrain Leading or Trailing attributes to Left or Right attributes.

StackView

- AutoLayout없이 View를 자동배치

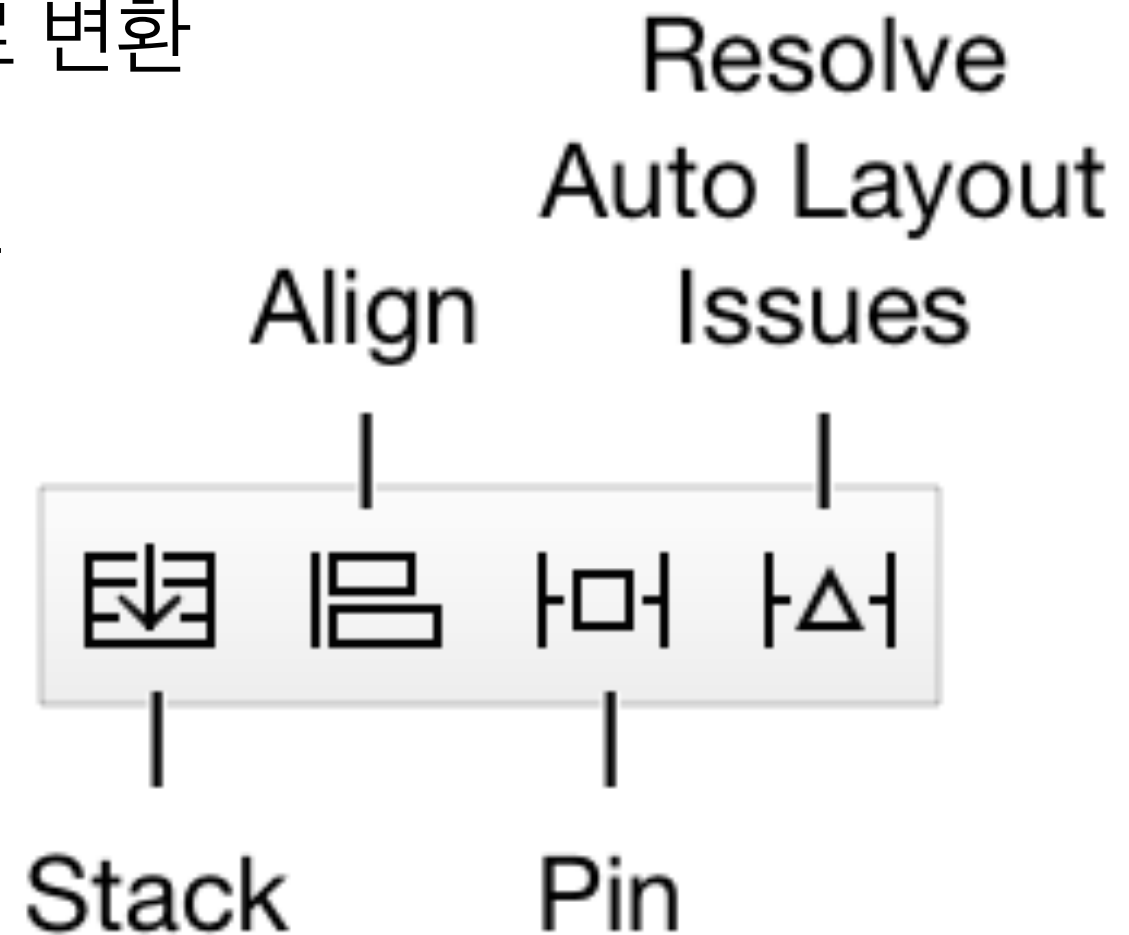


제약사항 만들기

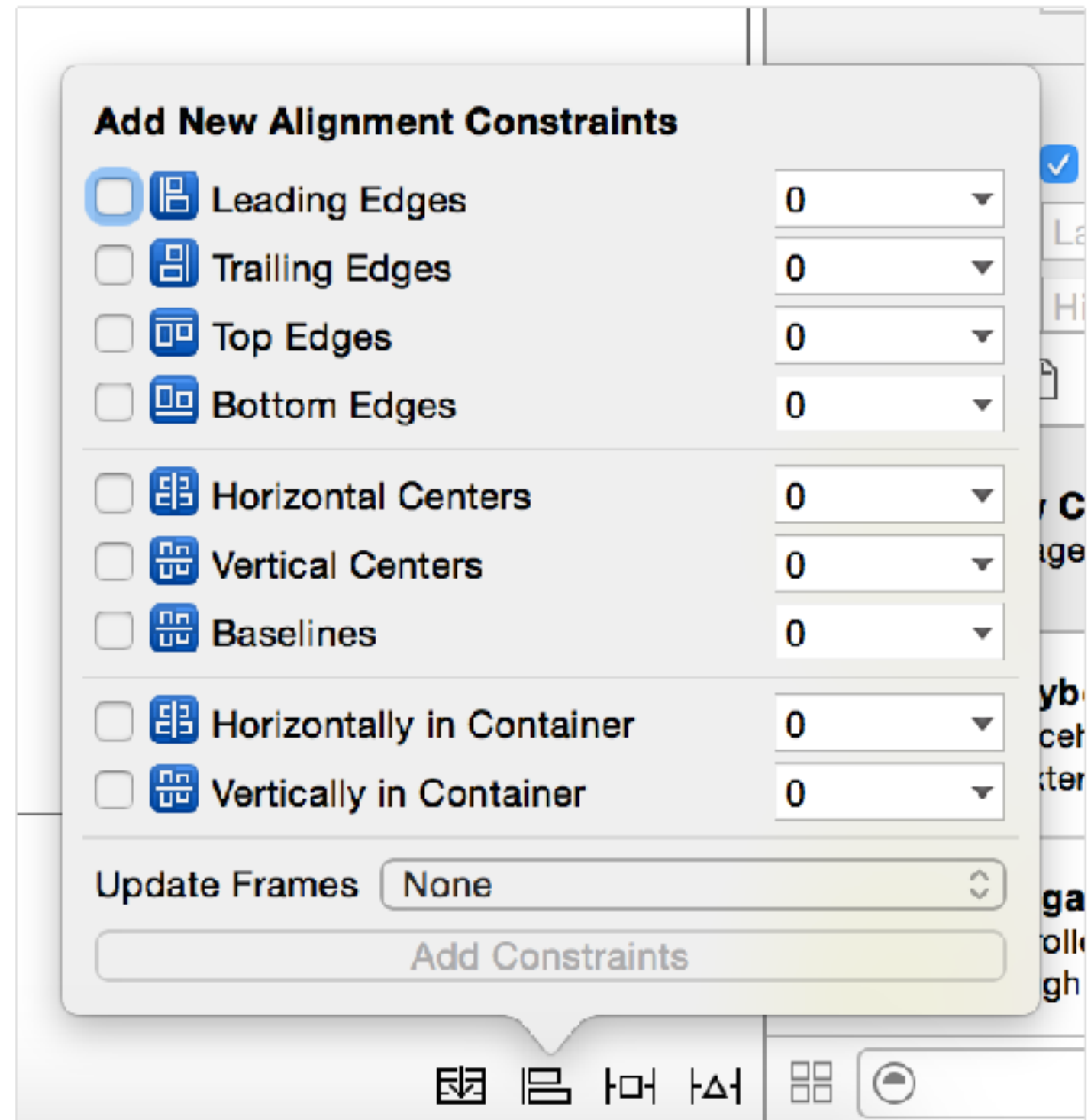
강사 주영민

AutoLayout Menu

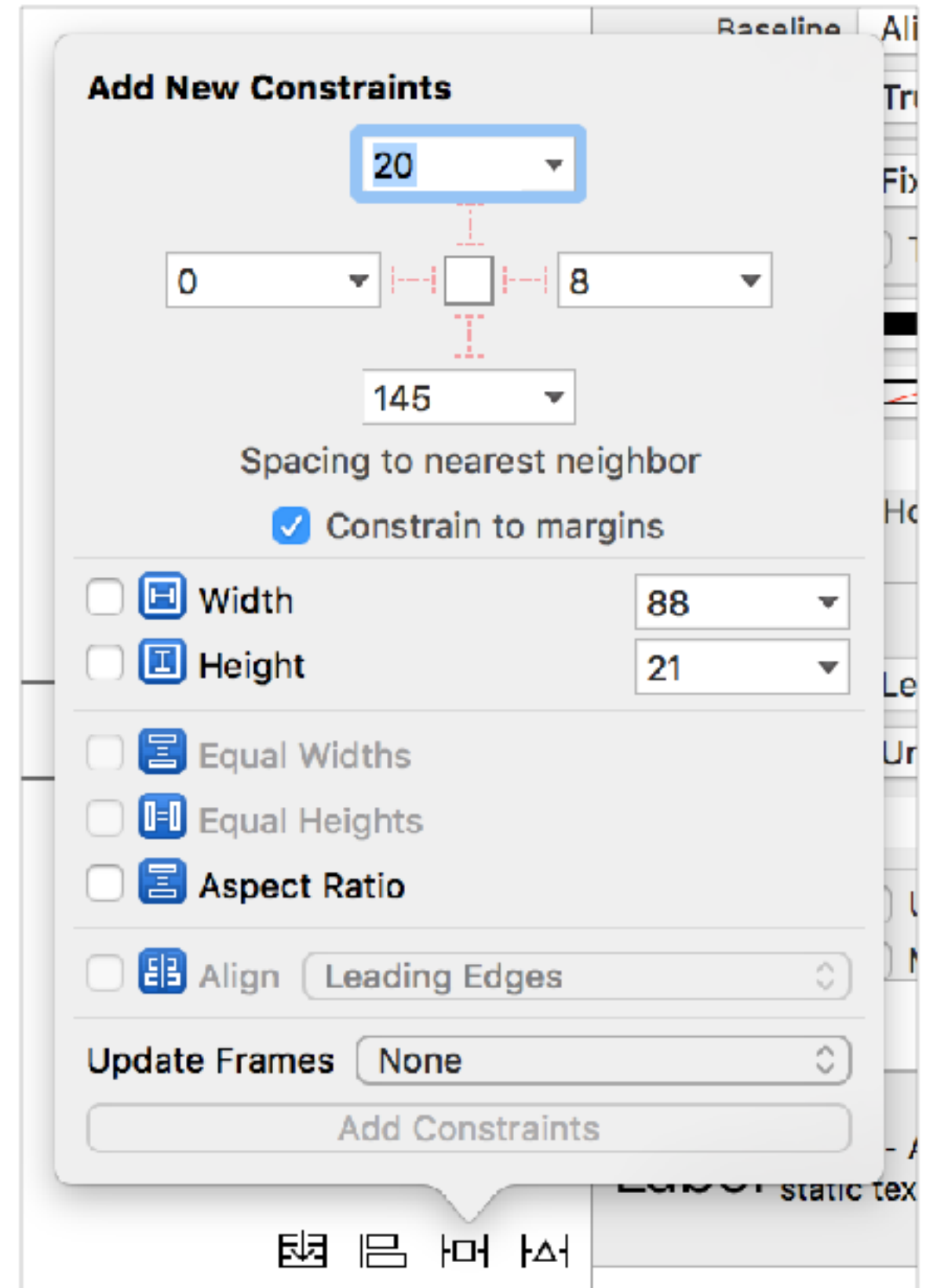
- Stack : 해당 객체들을 하나의 스택 뷰로 변환
- Align : 객체 정렬에 관한 제약사항 추가
- Pin : 객체의 크기 및 객체 간 거리에 관한 제약사항 추가
- Resolve Issues : 오토레이아웃 관련 문제 해결



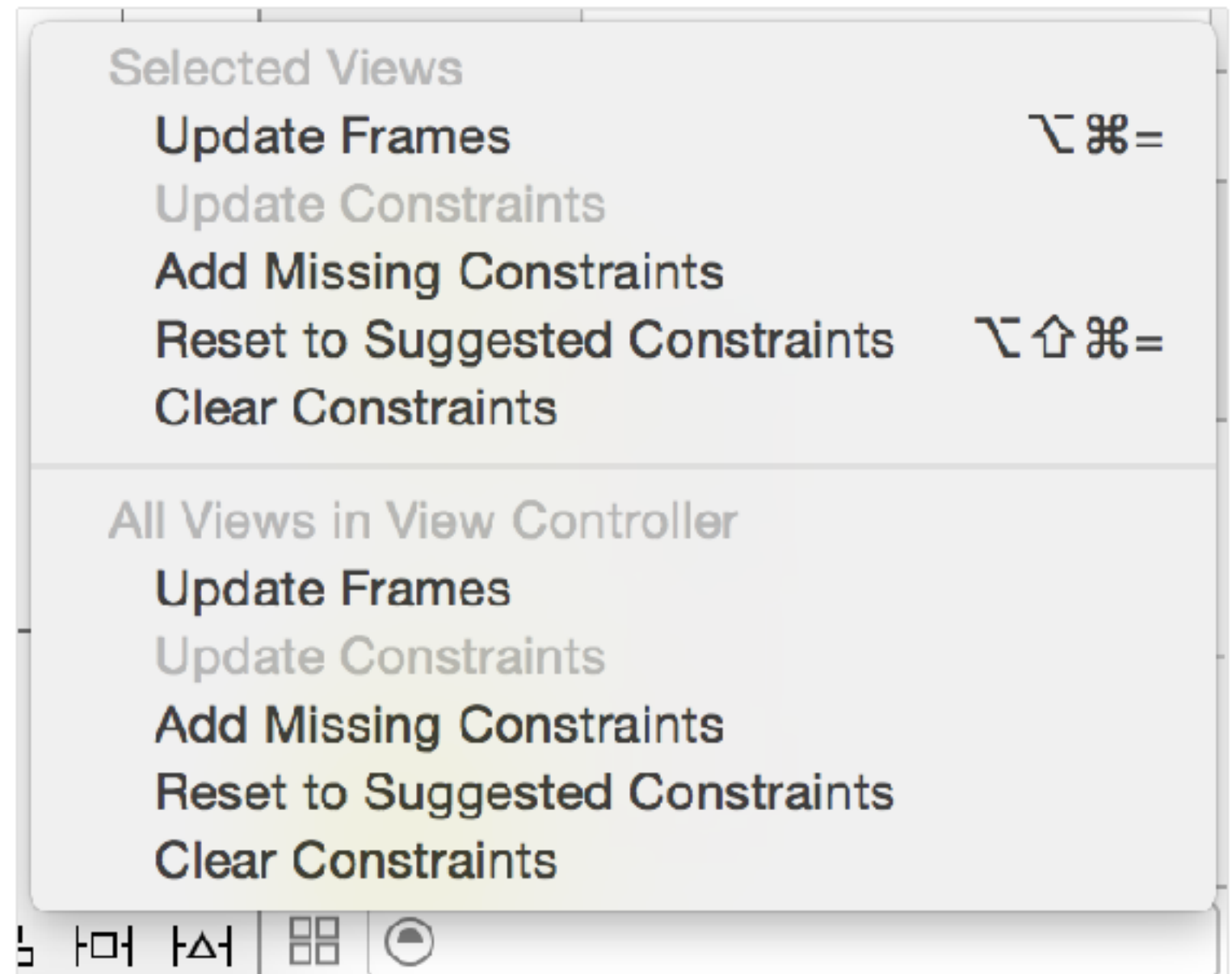
Align



Pin

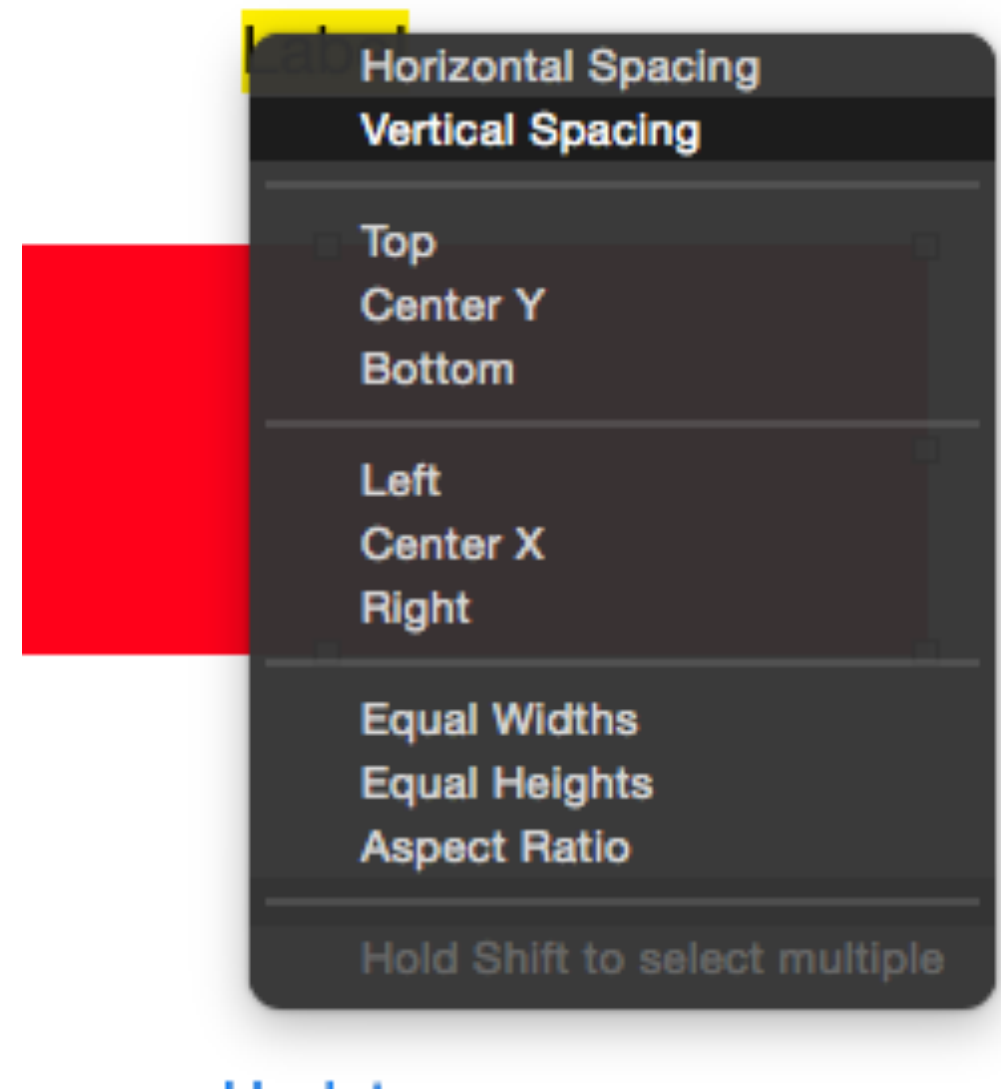


Issues



Ctrl + Drag

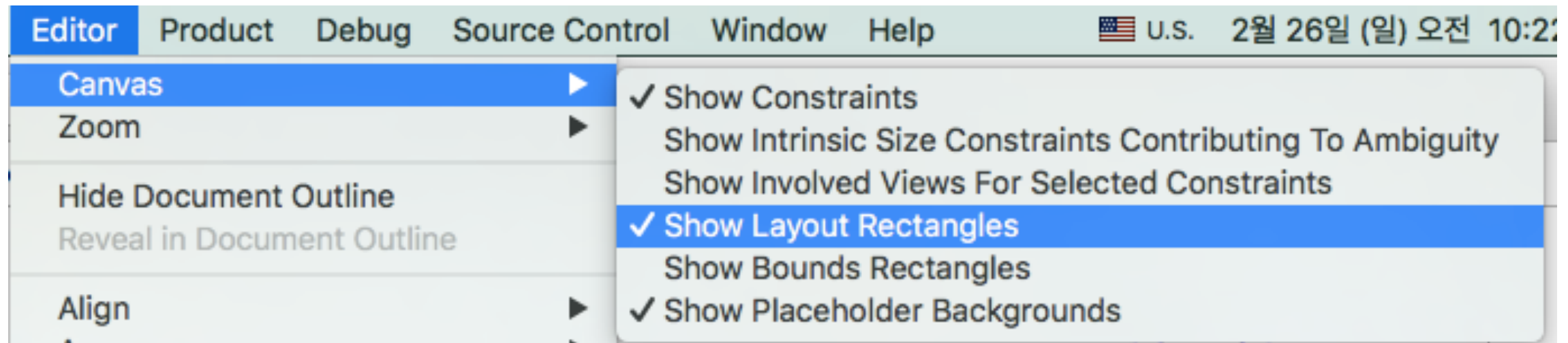
- 드래그의 위치와 방향에 따라 다른 제약 메뉴가 나타난다.



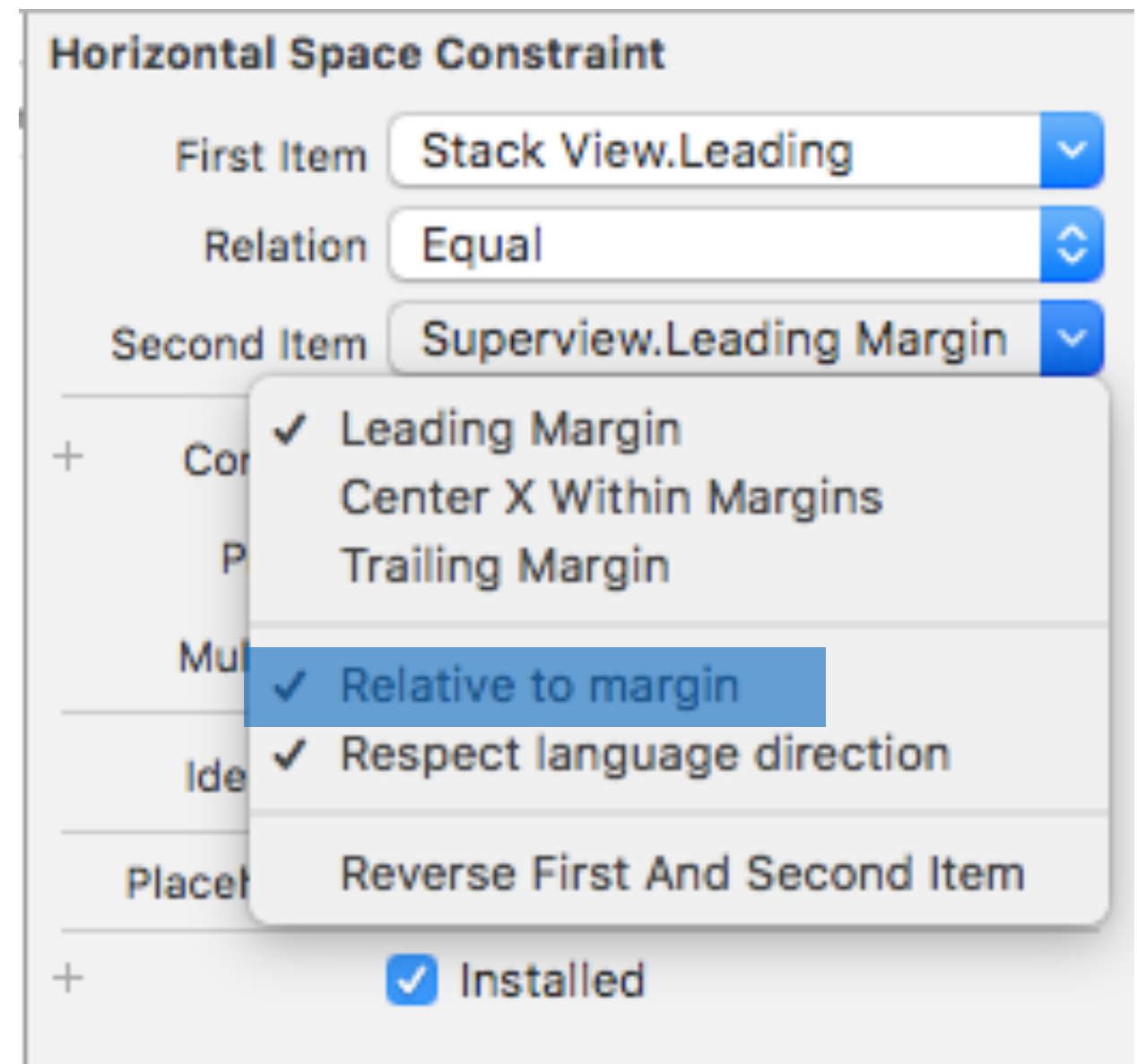
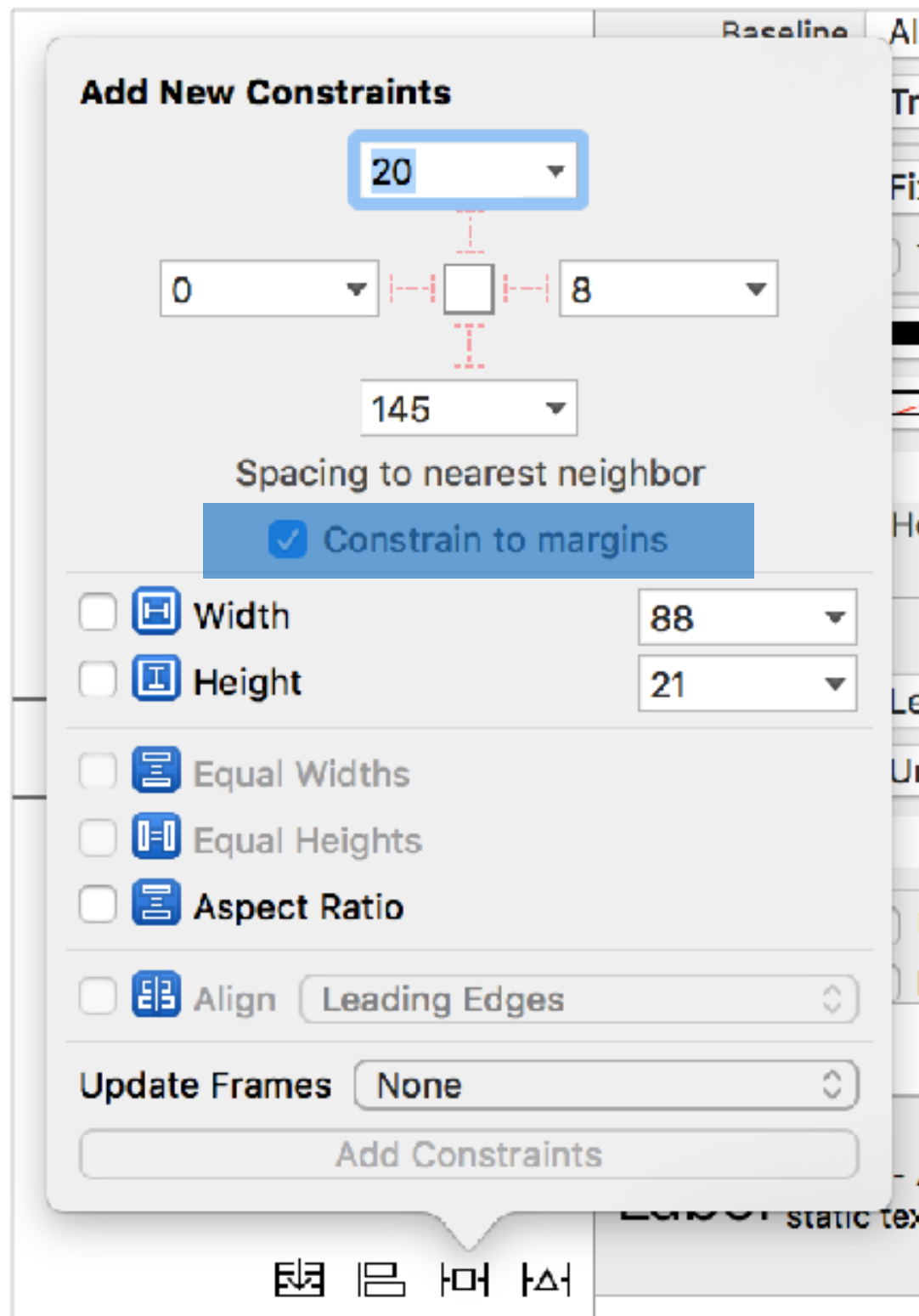
Layout margin



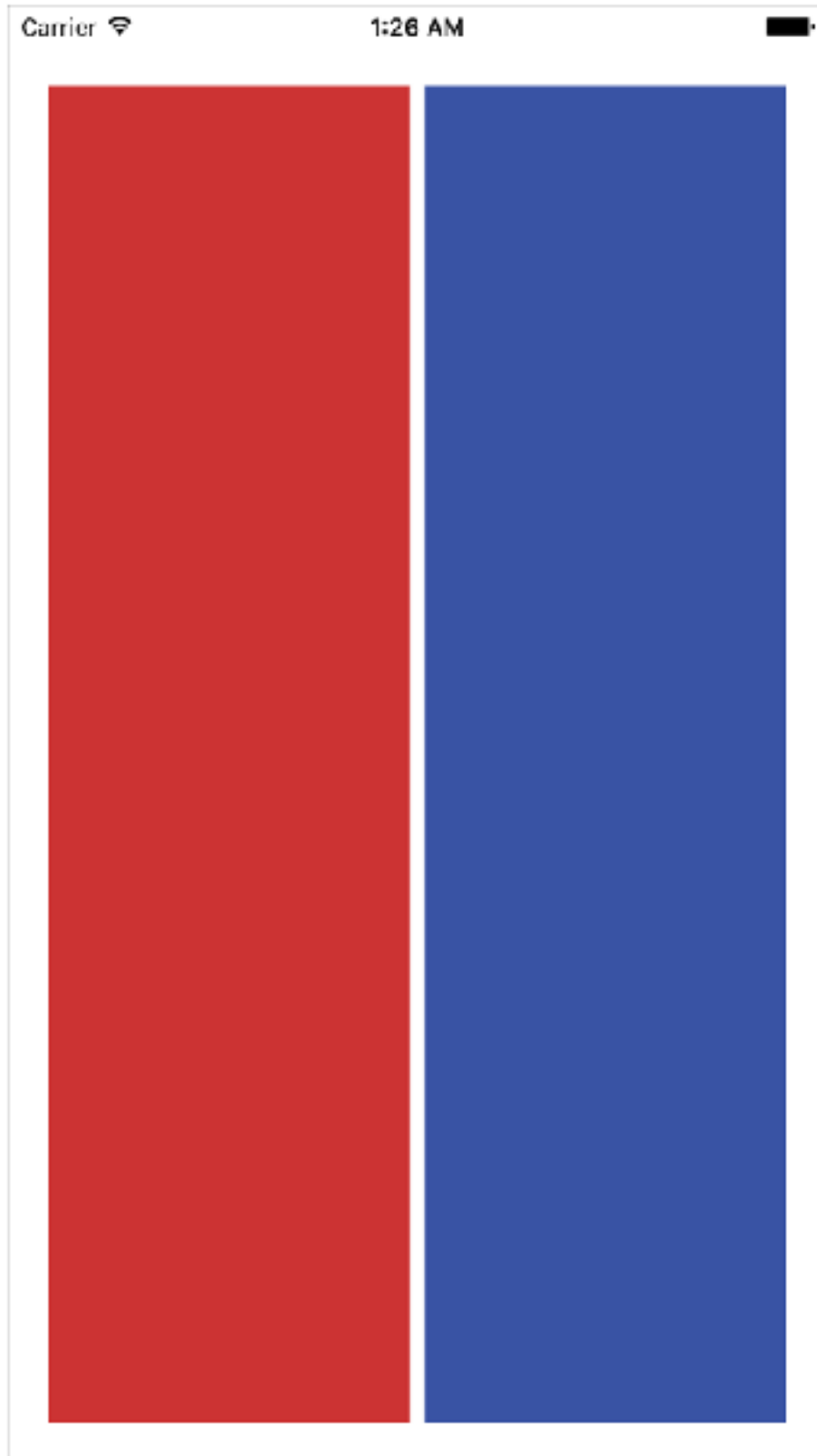
Show Layout Margin



Layout Margin 제거

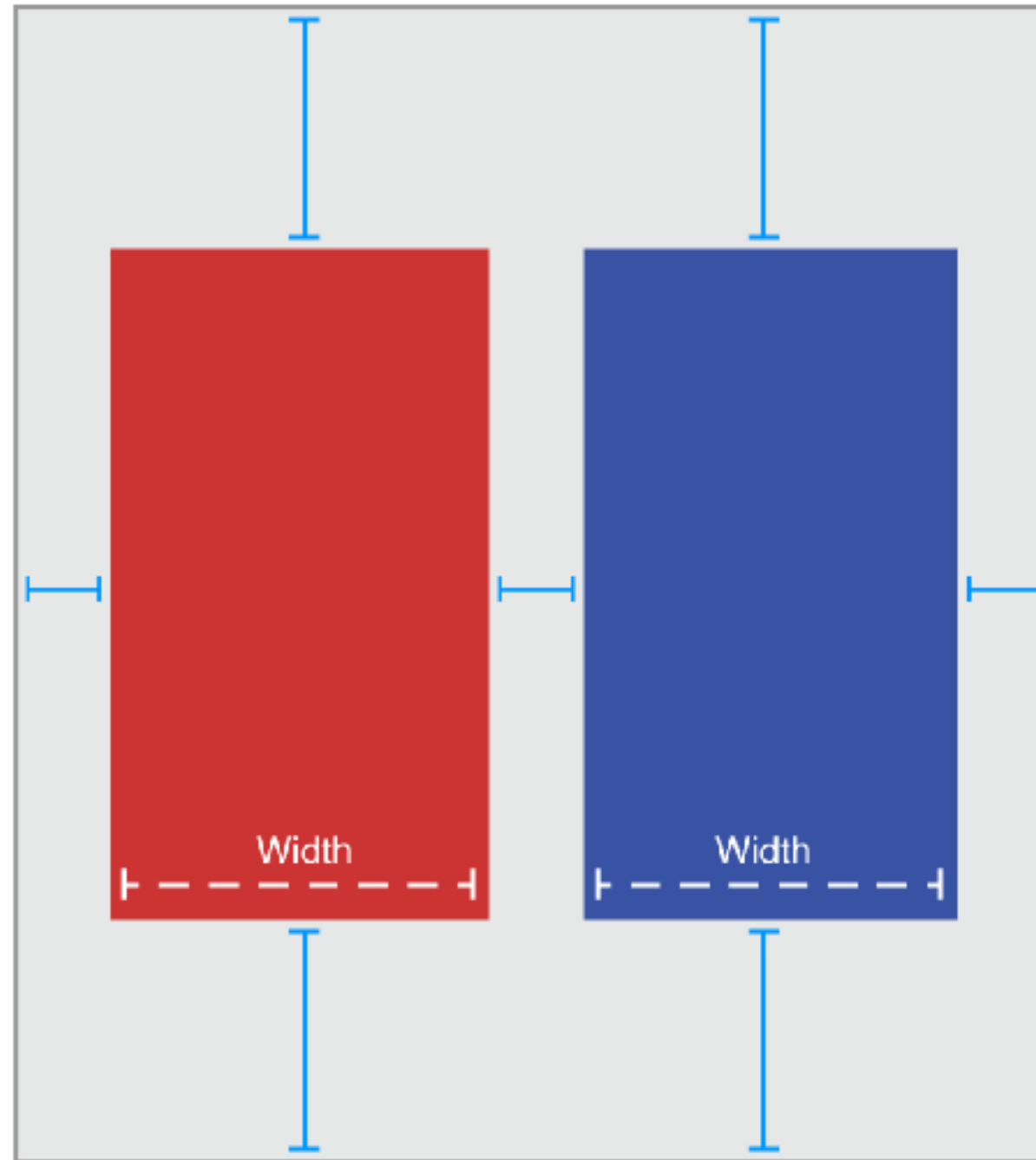


예제

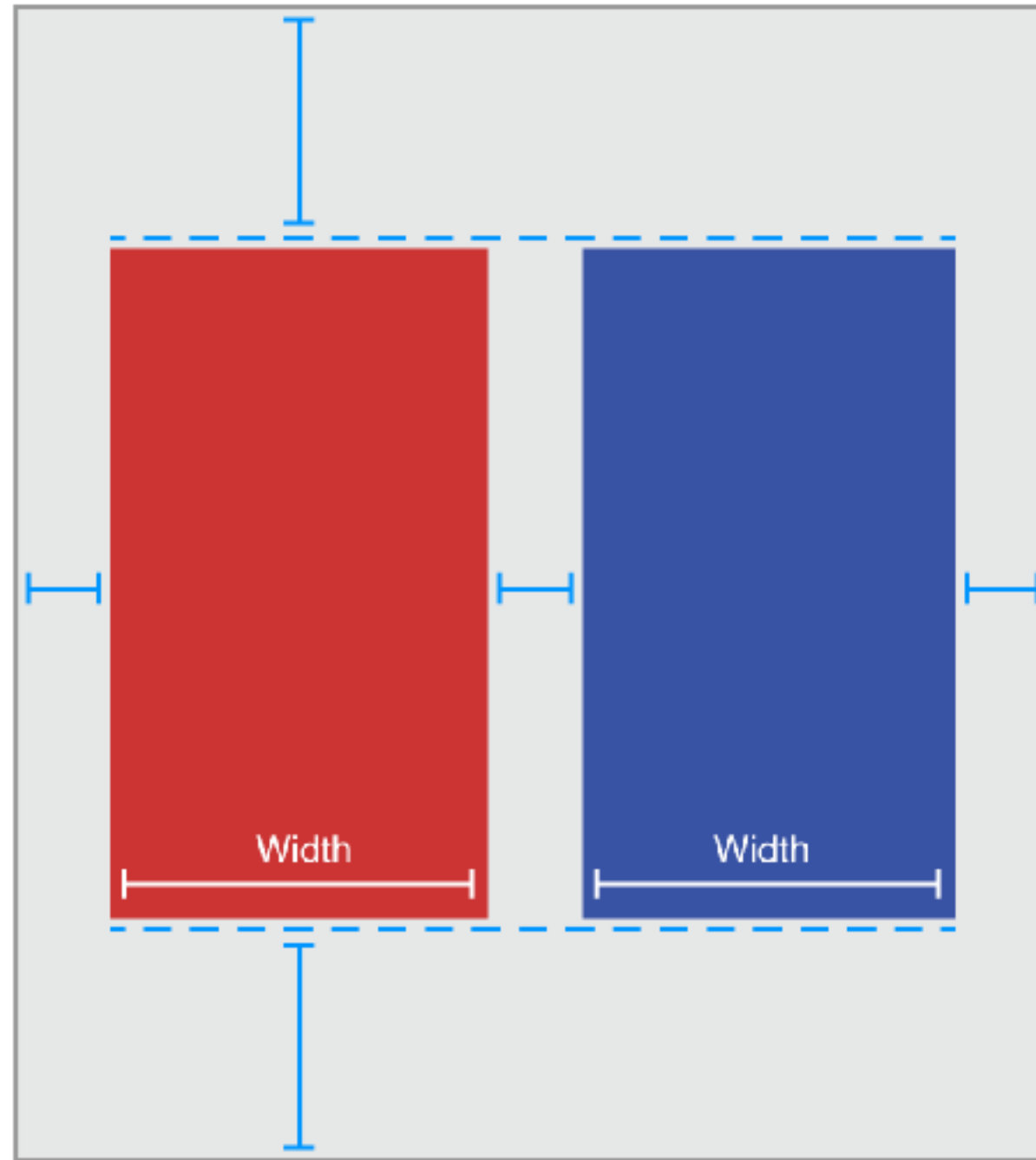


- 상, 하, 좌, 우가
Constrain margin
- 사이간격 5pt
- 뷰의 넓이가 같음

예제 - 해결책



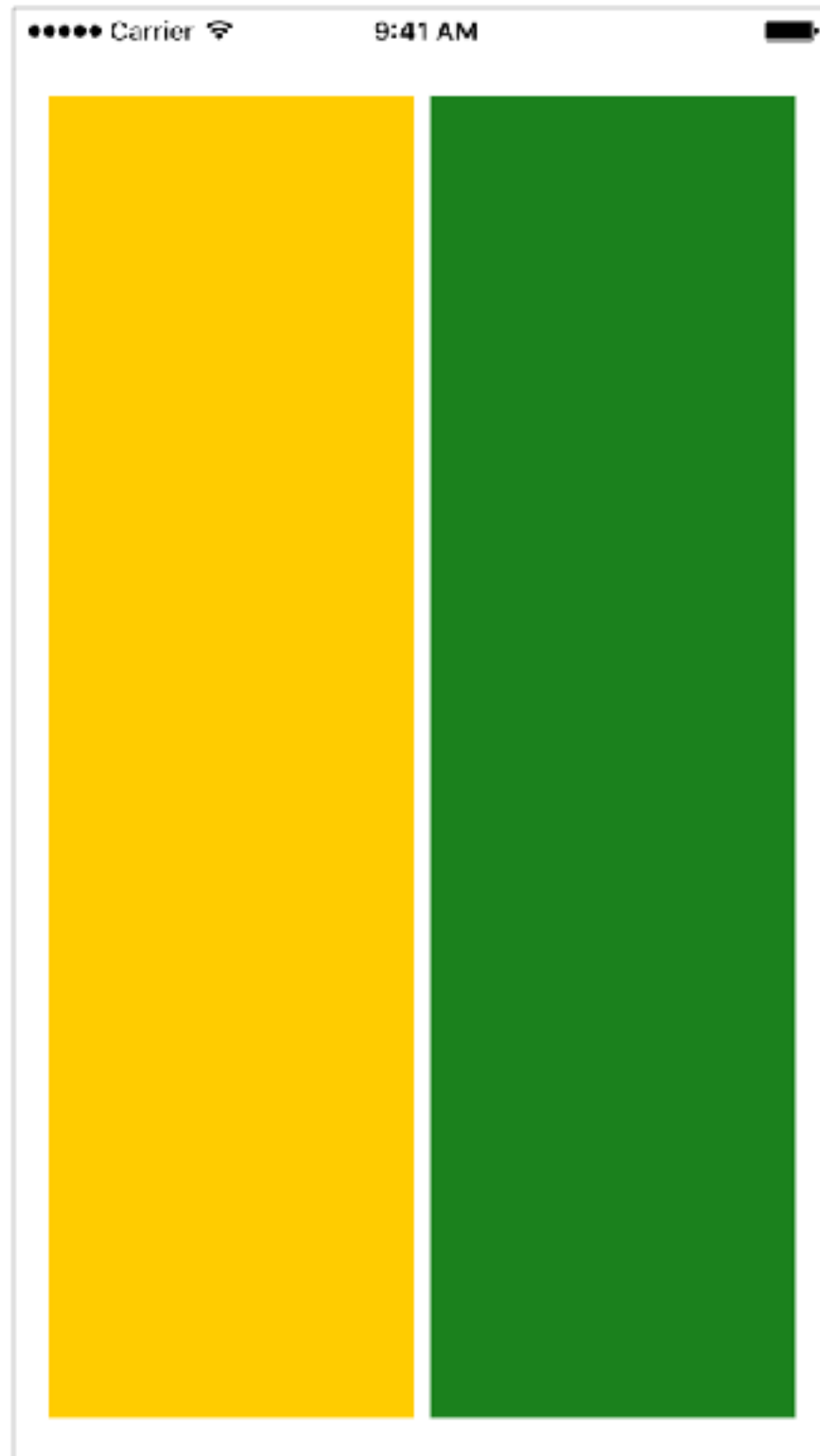
예제 -다른 해결책



Tip

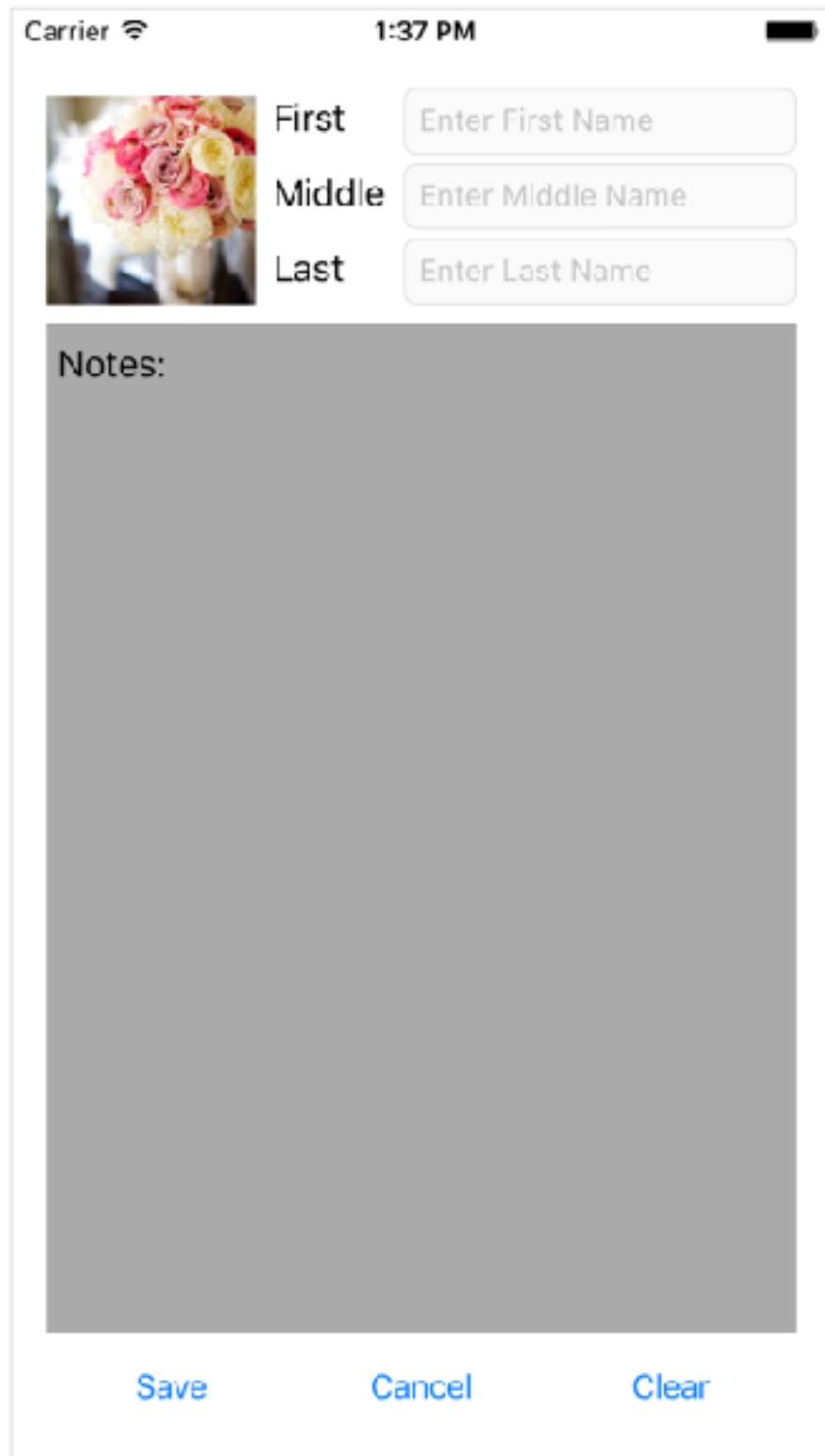
- 화면 배치의 기준이 되는 뷰를 잡고 시작하는 것이 좋아요!!

실습



- 상, 하, 좌, 우가
Constrain margin
- 사이간격 5pt
- 뷰의 넓이가 같음

실습



Carrier 1:37 PM

First Enter First Name

Middle Enter Middle Name

Last Enter Last Name

Notes:

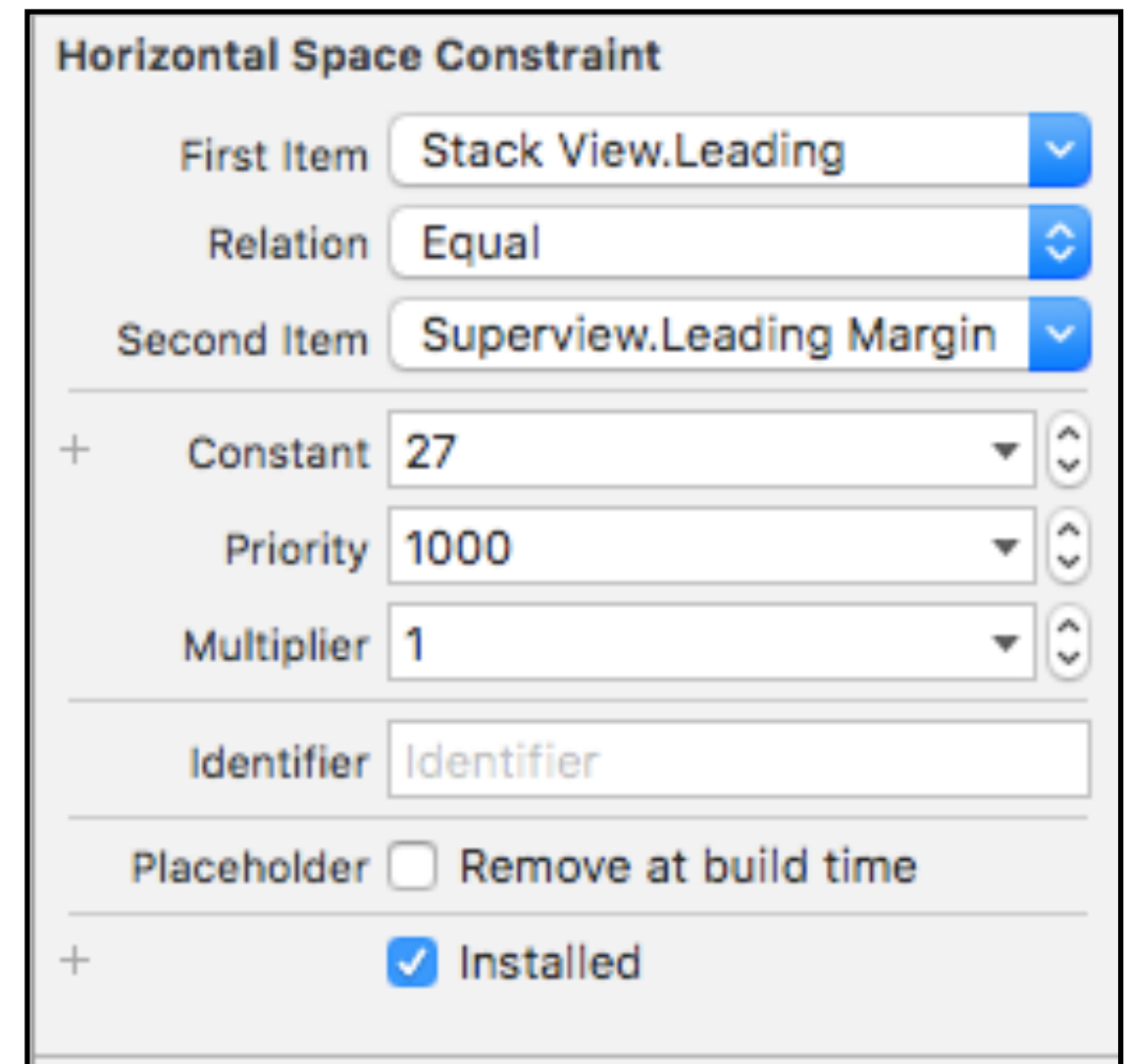
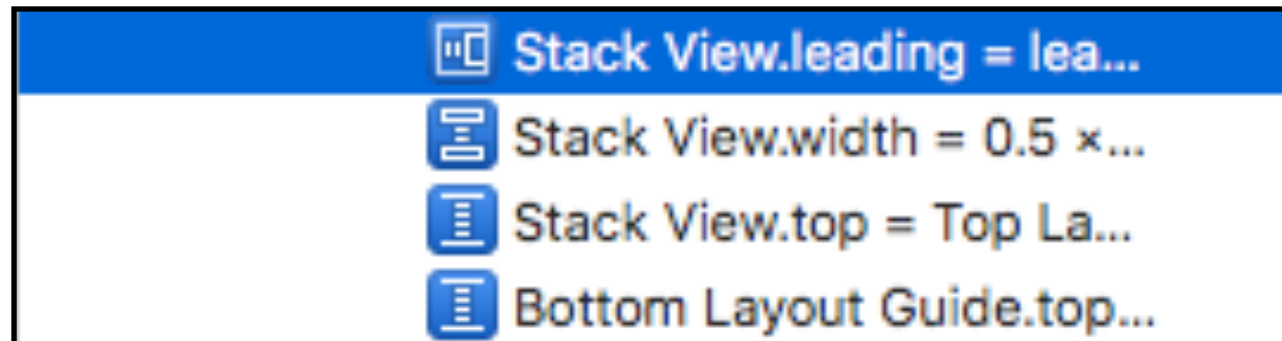
Save Cancel Clear

- 어떻게 만들어야 할까요?

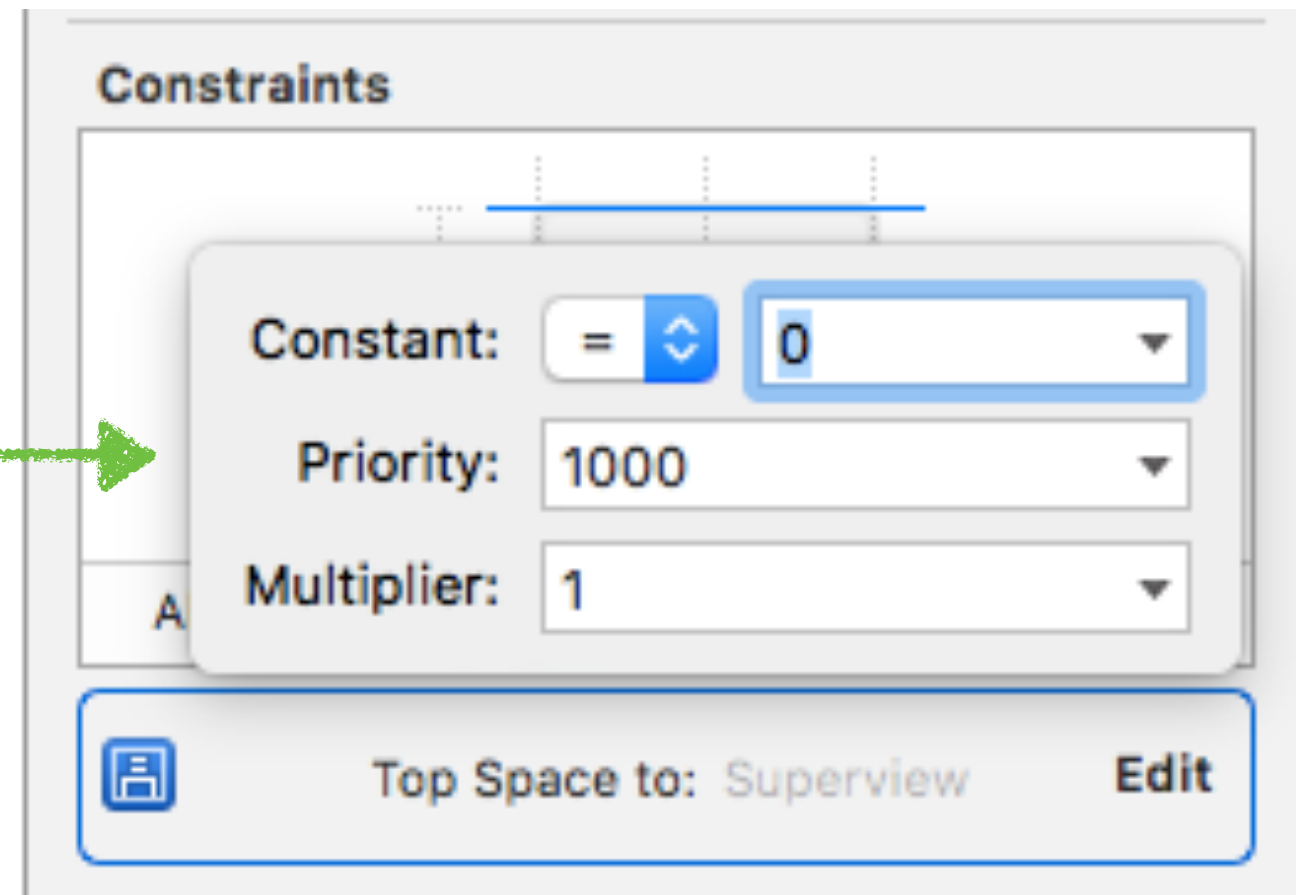
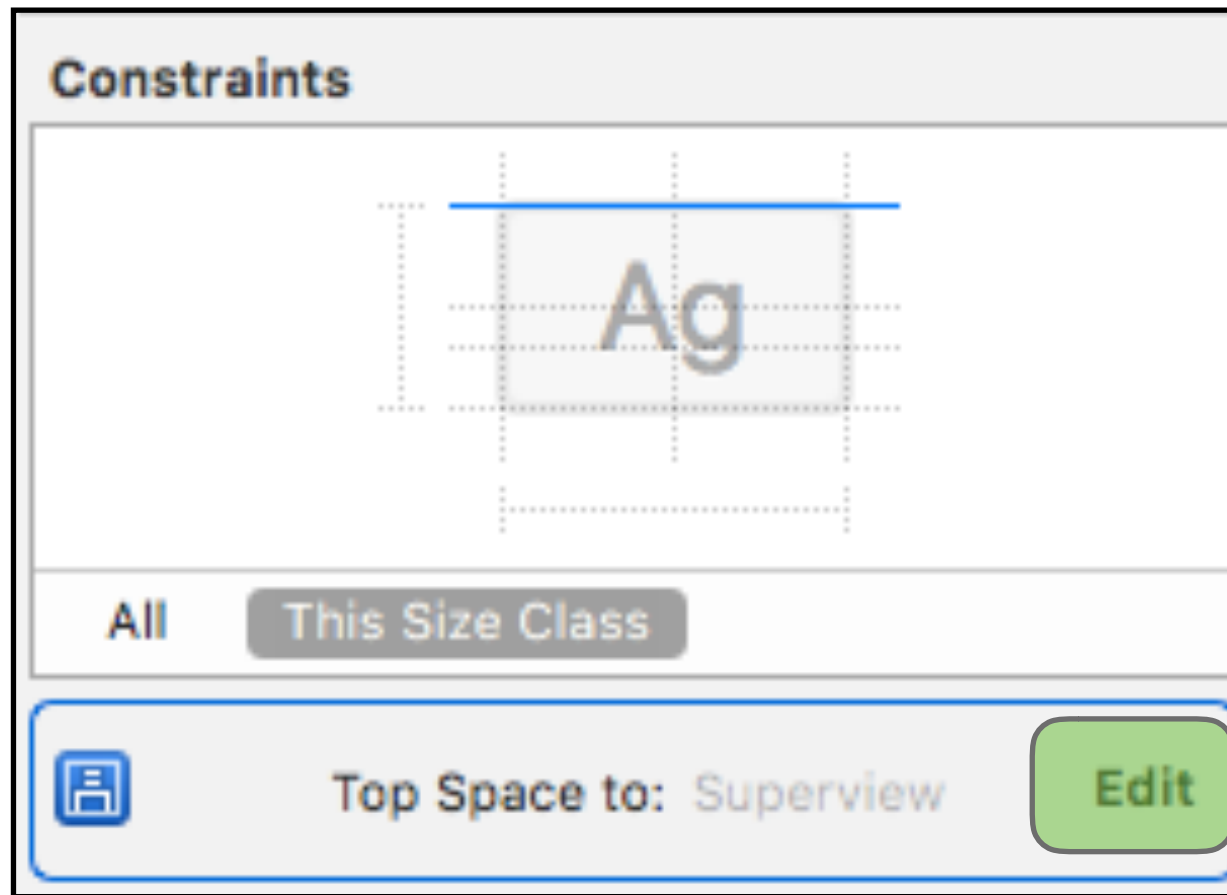
제약 설정

강사 주영민

constraint 선택 설정



constraint Edit 버튼 설정



설정 방법

Horizontal Space Constraint

First Item: Stack View.Leading

Relation: Equal

Second Item: Superview.Leading Margin

+ Constant: 27

Priority: 1000

Multiplier: 1

Identifier: Identifier

Placeholder ☐ Remove at build time

+ ☒ Installed

$\text{Item1.Attribute} = \text{Multiplier} \times \text{Item2.Attribute} + \text{Constraint}$

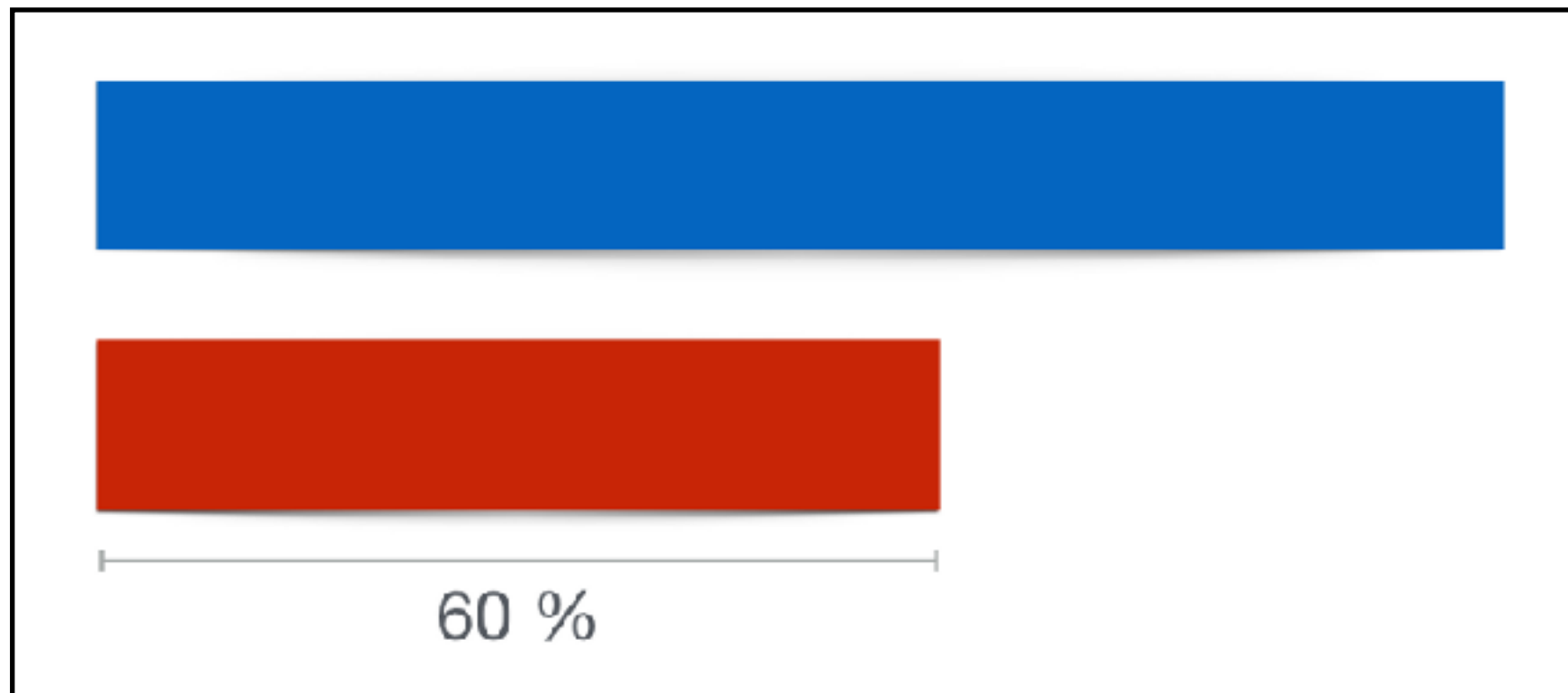
* 같은 Priority에서 같은 제약이 존재 할수 없다.

실습

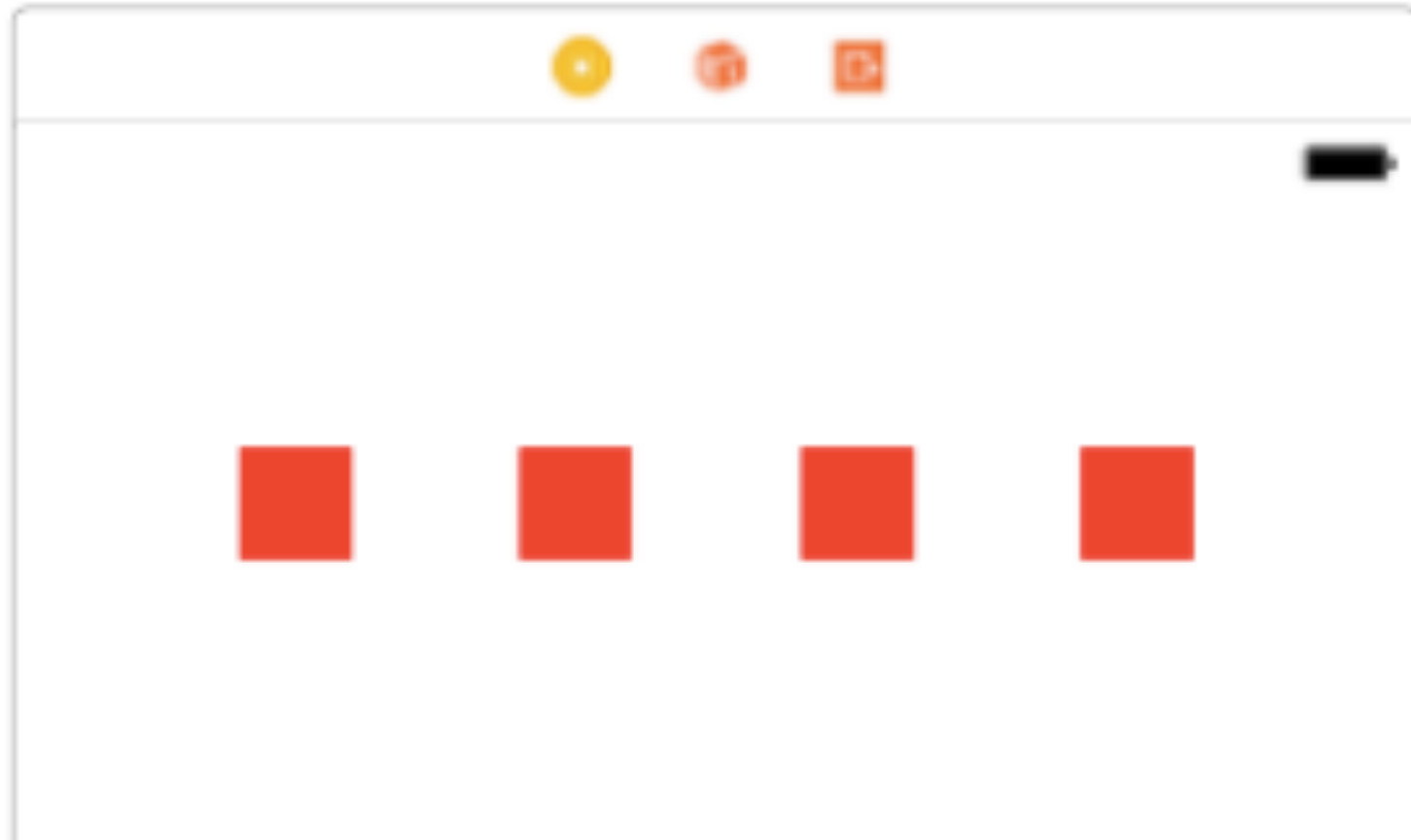


- 오른쪽 뷰의 넓이가 왼쪽 뷰의 넓이의 두 배인 화면

실습

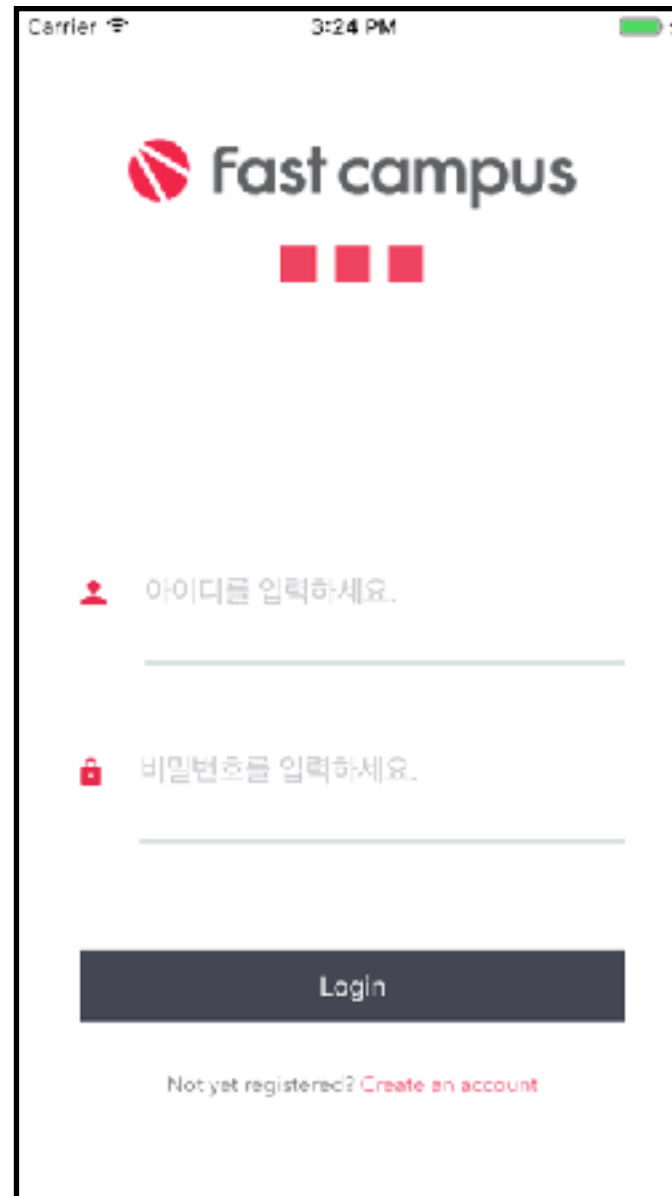


실습 : 동일한 간격 View



*StackView 사용 안하기


로그인 화면




A mobile app login screen for 'Fast campus'. The screen features the app's logo at the top, followed by three red squares. Below these are two input fields: one for the ID (labeled '아이디를 입력하세요.') and one for the password (labeled '비밀번호를 입력하세요.'). A dark blue 'Login' button is positioned below the input fields. At the bottom, there is a link for users who are not yet registered: 'Not yet registered? [Create an account](#)'.

Carrier 3:24 PM

 **Fast campus**



 아이디를 입력하세요.

 비밀번호를 입력하세요.

Login

Not yet registered? [Create an account](#)

UI

Delegate

Protocol

- 프로토콜은 원하는 작업이나 기능을 구현되도록 메서드, 프로퍼티 및 기타 요구 사항의 청사진을 정의합니다.
- 클래스, 구조체, 열거형은 프로토콜을 채택하여, 정의된 요구사항을 구현해야 됩니다.

Protocol 문법

```
protocol SomeProtocol {  
    // protocol definition goes here  
}  
  
struct SomeStructure: SomeProtocol, AnotherProtocol {  
    // structure definition goes here  
}  
  
class SomeClass: SomeSuperclass, SomeProtocol, AnotherProtocol {  
    // class definition goes here  
}
```


Protocol 문법

```
protocol SomeProtocol {  
    // protocol definition goes here  
    func someMethod(_ inputStr:String);  
}
```

```
class SomeClass: SomeSuperclass, SomeProtocol {  
    // class definition goes here  
    func someMethod(_ inputStr:String)  
    {  
        print(inputStr)  
    }  
}
```

Delegate Design pattern

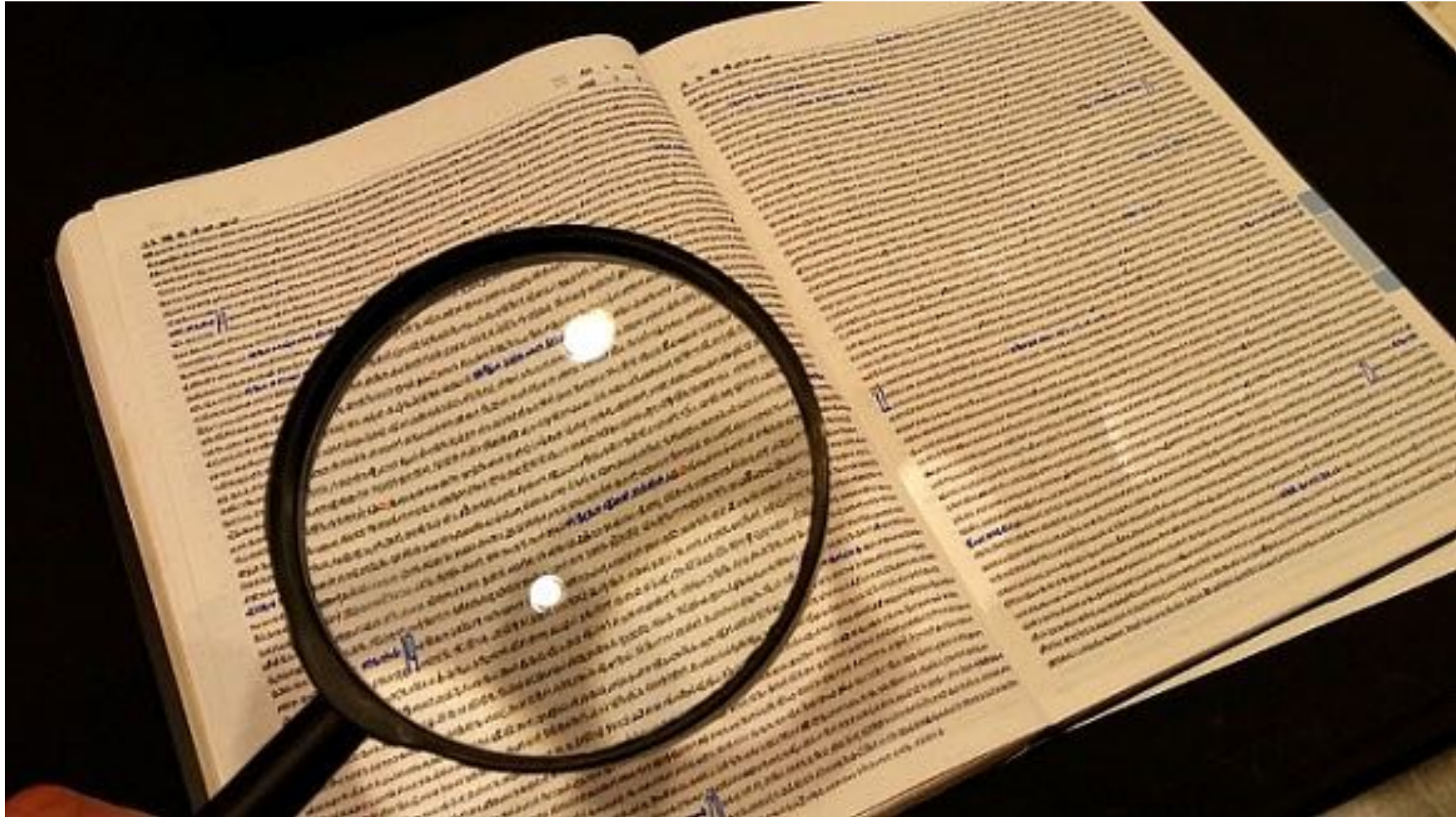
- Delegation is a design pattern that enables a class or structure to hand off (or delegate) some of its responsibilities to an instance of another type.
- 델리게이트는 클래스나 구조체에서의 일부분의 할 일을 다른 인스턴스에게 대신 하게 하는 디자인 패턴!

UITextField

- 사용자 텍스트 입력을 위한 UI Component.



UITextField



TextField Delegate 확인

- Delegate 확인하기

UITextField - 실습



텍스트 입력 확인

결과 : 텍스트 입력

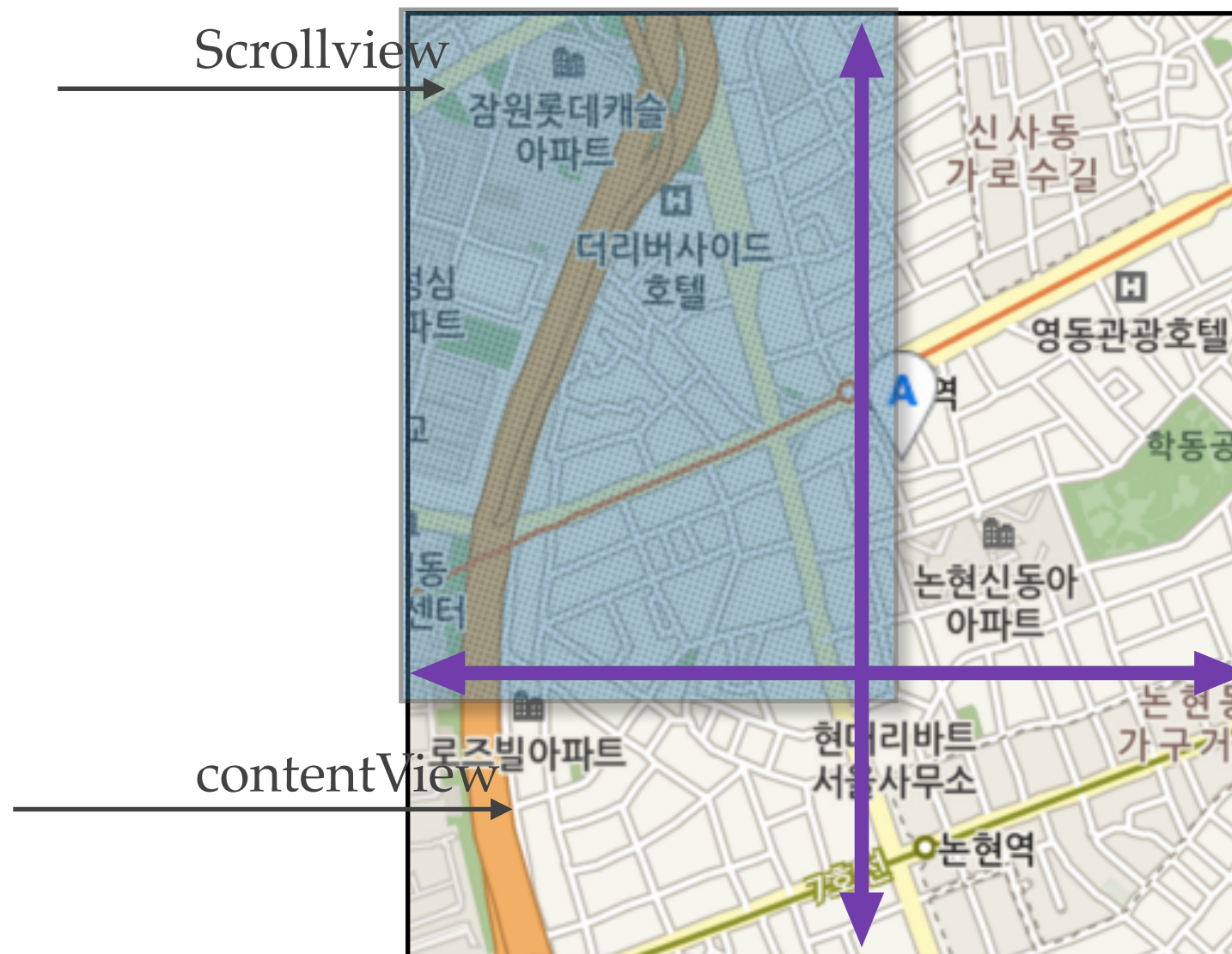
1. 텍스트 필드에서 텍스트 입력
2. 확인 버튼 클릭
3. 아래 레이블에 입력된 텍스트 보여주기

UIScrollView

- ViewSize보다 확장된 뷰를 보기위한 View
- UIScrollView에 추가된 View는 ContentView위에 추가 된다.

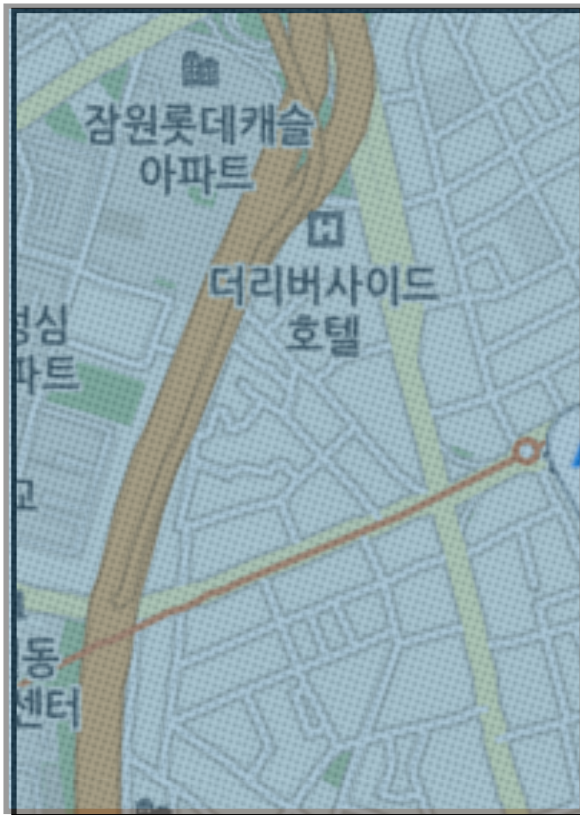


UIScrollView - ContentView



UIScrollView - 스크롤

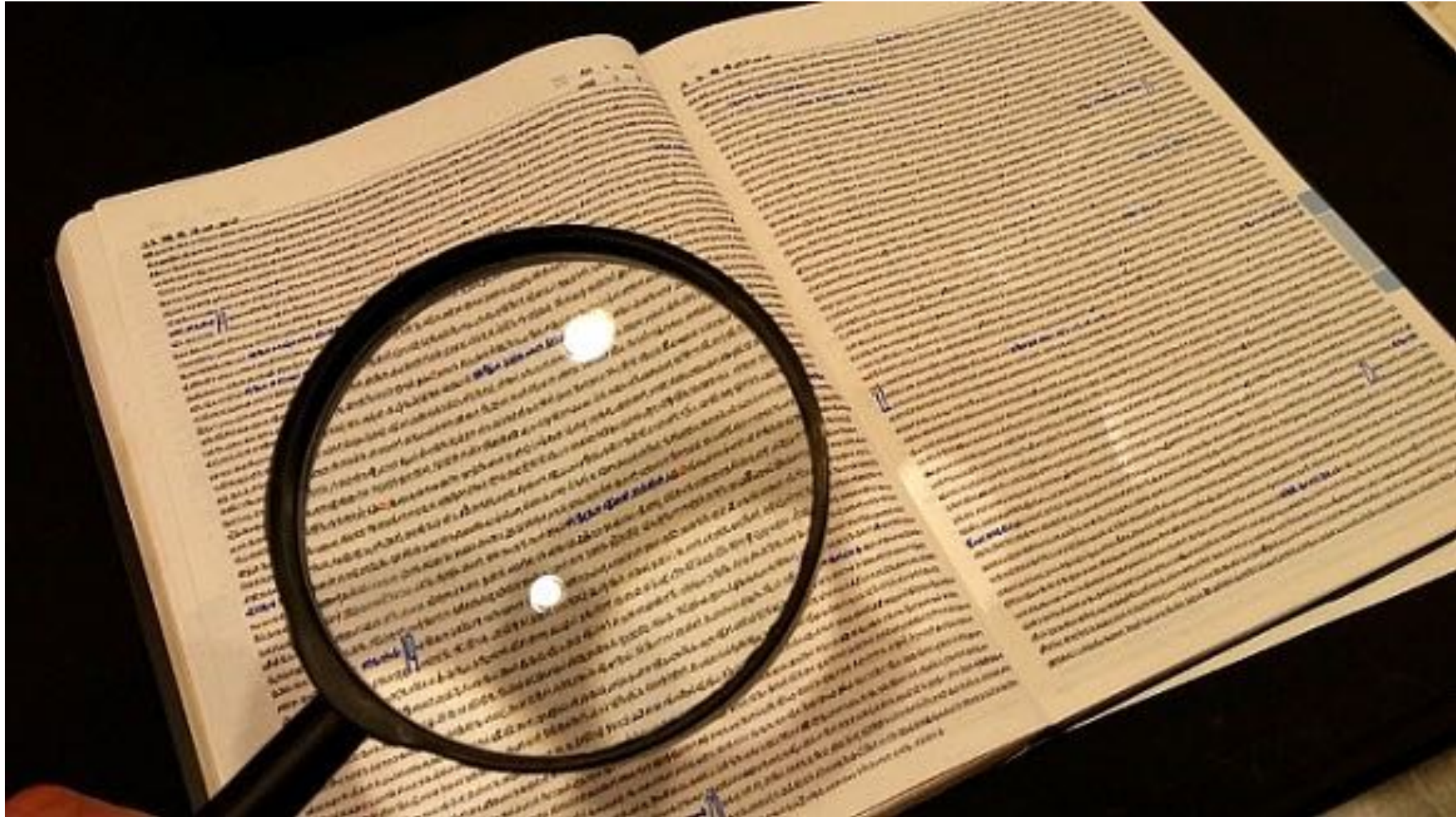
실화면



내부동작



UIScrollView

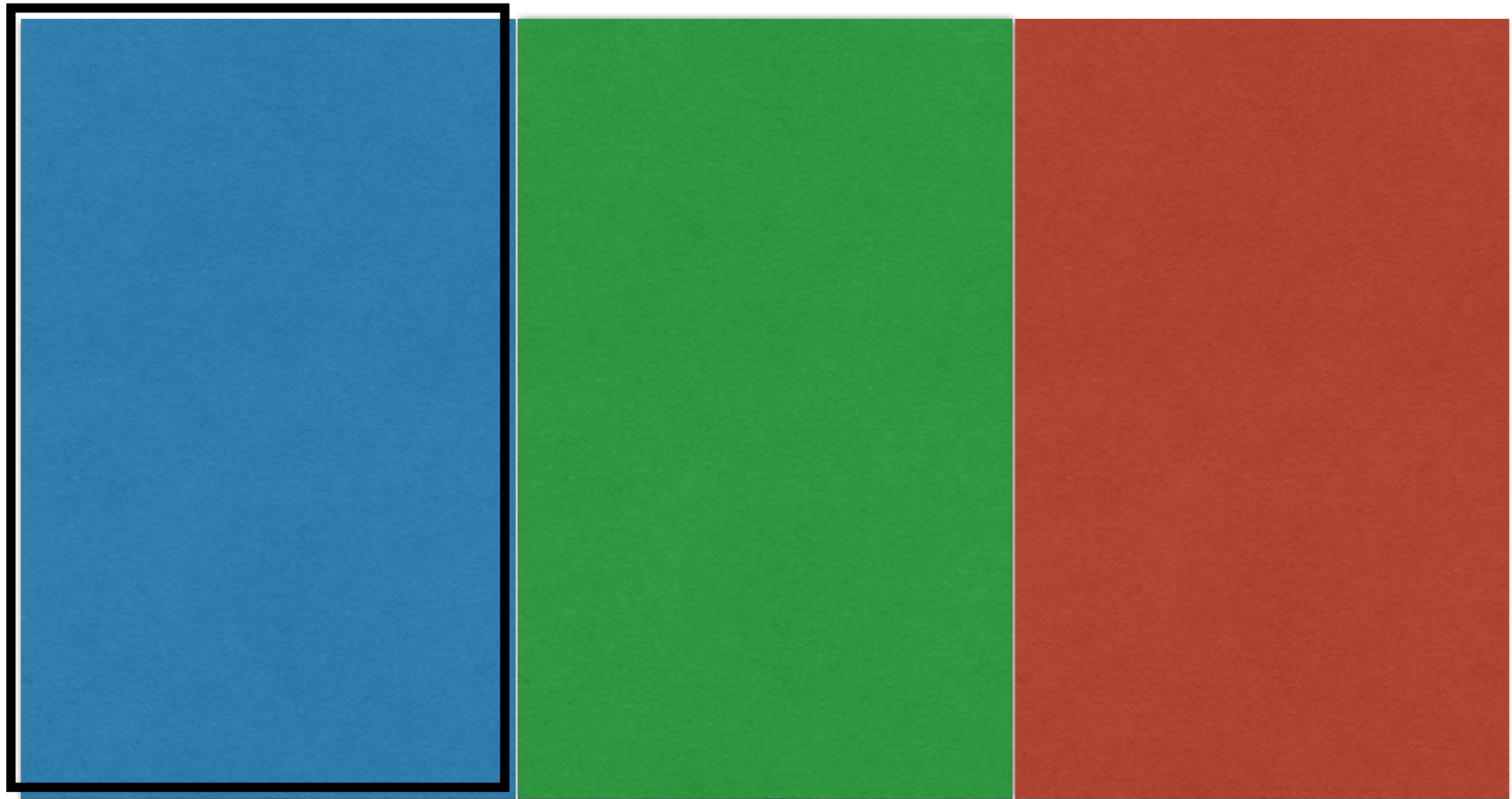


UIScrollView 만들기

1. 스크롤뷰 추가 (Autolayout 지정)
2. 콘텐츠 뷰 추가하기
 - UIView 추가
 - add top, bottom, leading, and trailing edges
3. (선택 사항) 가로 스크롤을 사용하지 않으려면 콘텐츠 뷰의 너비를 스크롤보기의 너비와 같게 설정합니다.
4. (선택 사항) 수직 스크롤을 사용하지 않으려면 콘텐츠보기의 높이를 스크롤보기의 높이와 같게 설정하십시오.

UIScrollView - 실습

가로 3페이지 스크롤 뷰 만들기(컨텐츠는 이미지뷰)



ScrollView Delegate 확인

- Delegate 확인하기